附加说明

总体评价

本次代码最大的优点是将主机激活步骤在代码层面上进行了解耦，各个激活步骤在代码层面上的依赖已经很小，未来在步骤增减和先后顺序变更时，对代码修改的范围确定且相对独立。另外，代码实现上对未来为不同类型主机进行激活时预留了实现的空间。代码可维护性和扩展性有了较大提升。

激活步骤在业务层面上灵活性则还有提升空间，目前在激活时执行的命令有许多方面是可以配置的，比如路径等，但是还不够彻底，命令中还有很多部分是硬编码的。更进一步地，还可以将各步骤的执行顺序也写到配置中，这样，未来对代码的修改需求会更小。

在代码设计方面，类 CKeyMainWnd 的职责太多，UI 和激活步骤都实现在该类中，应该进行拆分。拆分后，如果未来要实现命令行激活工具，实现激活步骤的类可以直接复用。

在细节方面，有不少不规范的地方或者小问题，部分是正式编码规范出来之前老代码遗留的，部分是新引入的。

另外，如果编译器支持且鼓励使用 C++11 标准的话，有些地方实现起来会更简单、灵活、优雅。

一些其他建议

1. KeyMainWnd.h:19：typedef 放在类中更合理，外部并不需要知道它的存在。
2. KeyMainWnd.h:52：使用 vector 即可，无需使用 deque。在使用中没体现出作为队列的特征。
3. KeyMainWnd.h:53：uint32_t 使用枚举代替更好。
4. 对宏的使用多进行些思考（其他代码：CHECK_RETURN、CHECK_BREAK 等）。定义并使用某个宏带来了理解便利还是理解负担，提供了多少抽象，有多少可能会造成误解。在确实值得使用宏的时候才使用。

C++11 标准应用推荐

std::function

使用 std::function 对象作为回调参数，std::function 统一了 C++中的可调用对象，包括函数指针、函数对象（重载 operator()）、lambda 表达式、函数方法指针。这样，调用回调的一方只需要知道它的调用签名，而不用管它具体是什么类型。如：

```
E_RetCode CKeyMainWnd::Active_Step3()
{
    // …
```

```
        return m_ptrKeySshUdExec->
            ExecCmd(szCmd, Name_szCmdExecDir,
                    std::bind(&CKeyMainWnd::OnNextStepCallback, this, _1, _2, _3, _4);
    }
```

auto

定义迭代器变量时，使用 auto，让编译器进行类型推断。这里推断的类型非常明确，不会造成歧义也不属于滥用。如：

```
auto iter = m_mapActiveStepFun.find(m_curStep);
```

类内初始化

```
CSshUDExec *m_ptrHostSshUdExec = nullptr;
```

其他

使用 std::chrono 库进行时间相关的操作。
使用 std::begin/std::end 对数组进行迭代。